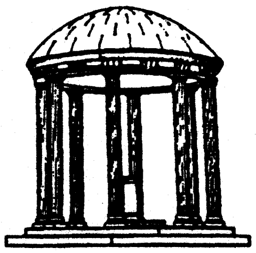
A semantics for priority using partial orders with a simultaneity relation

TR90-008

February, 1990

Peter H. Mills



The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

A semantics for priority using partial orders with a simultaneity relation

Peter H. Mills

Department of Computer Science, University of North Carolina
Chapel Hill, N.C. 27599-3175 USA
E-mail: phm@cs.unc.edu

Abstract

In this paper we present a semantics for concurrent systems with priority constraints, using as a basis partial orders of events with both precedence and simultaniety relations. Previous work has shown that theories based on partial orders modelling precedence alone are insufficient to capture priority. We use the *prosset* model of Gaifman and Pratt, an extension to partial-order theories which includes a relation for explicity simultaneity, to develop a semantics for a COSY dialect allowing the specification of priority. We show that Janicki's "sequences of multiples" semantics for priorities can be derived from a linearization of the prosset semantics. Finally, we look at a simple taxonomy to clarify the relation of these various semantics to other event-history models.

1. Introduction

1.1. Priority

In the context of concurrent systems, priority denotes the order of preference in which conflicting events obtain service. This preference can depend upon the nature of the requested service or the importance of the events. Priority is widely held to be a basic concept whose specification in concurrent system description serves as the mechanism to ensure such properties as fairness and absence of starvation [Lamp84]. Techniques for specifying priority in concurrent system description abound: in real-time systems [Quir85], in occam [Inmos87], and in COSY [Laue79]. To reason about such specifications requires a formal semantics. In this paper we develop such a model-theoretic semantics for COSY which provides a more elegant alternative to previous theory.

1.2. COSY

COSY is a notation for concurrent systems which specifies the synchornization properties of the system in abstraction from interpretation of the composite events [Laue79]. COSY grew from the concept of path expressions as synchronization mechansism [Camp74], was introduced as a specification notation in [Laue75], and later extended in [Laue79].

Briefly, a COSY *path program* is a collection of paths, each consisting of a regular expression extended with parallel operations [see figure 1]. The sequential subsystems described by each path operate in parallel, and are synchronized by common event names. The intent is for each path to express a constraint on the order of activations of events in it. The abstract syntax for path programs, as found in [Best86, Jani88], is exemplified in Figure 1. *Priority path programs* specify priority with a separator $<$ between actions, a notation introduced by Campbell in [Camp76]. $a < b$ means "if both a and b might occur, then choose b": this defines a fixed priority.

1.3. COSY semantics: partial order models

A standard semantics for path programs without priority constraints was developed by Shields using *vector firing sequences* [Shie79], which are equivalent to labeled partial orders of events [Best86]. Here, as in many linear-time semantics of concurrency, the denotation of a process is a set of *behaviors*, each representing one possible history of the computation, one possible observation. Each behavior consists of a *partial order* relation among events indicating precedence and causality. Such a behavior is given in Figure 2. The intended interpration of unrelated events in a behavior is that they may occur in either order or simultaneously, thus expressing true concurrency. This is in contrast to *total orders*, which reduce parallelism to arbitrary interleaving, and so cannot distinguish between nondeterminism and true concurrency. Partial order semantics have many advantages, as discussed by [Prat86].

1.4. Priority semantics: previous work

Janicki [Jani87, Jani88] showed that partial order models are insufficient to capture priority: the partial order describes more observations than are possibly allowed by priority constraints. Alternatively, he proposed a semantics where each observation is a sequence of sets of simultaneous events. This is similar in form to "step semantic" models based on the idea of using collections of simultaneous actions to model true concurrency [Taub87, Meye89]. The notion of step derives from Petri net theory, where subsets [Roze83] and multisets [Reis85a] of concurrently firable transitions are considered. The key to modeling priority is in expressing explicit simultaneity of events, beyond the true concurrency expressed by partial orders.

		Equivalent Notation
Priority	priority $b > c$ end	$b > c$
Path Program	path a;b end path b,c end	(a;b)* || (b,c)*

LEGEND: , mutually exclusive choice
; sequence
* Kleene closure

Figure 1: COSY path program

1.5. Priority semantics using partial orders with explicit simultaneity

We propose using an alternate model, the *prosset* (pre-order specification set) model of processes [Gaif87], as a basis for developing priority semantics. Here, a behavior consists of a set of events partially ordered by both precedence and simultaneity relations. We derive two semantics for priority using prossets: both refine the path program's vector-firing sequence (VFS) partial orders to remove conflicts with priority constraints. In the first, we *split* the VFS partial orders for path programs into a set of prossets augmented with relation arcs. In the second we *linearize* the VFS prossets, remove those observations with priority conflicts, then take the *maximum ideal partition* of the remaining set.

We show that Janicki's semantics for priority is equivalent to the prosset-linearization of both our semantics. Our formulation has the advantage of employing an existing elegant model with fairness results. Finally, we look at a simple taxonomy to clarify the relation of these various semantics to other event-history models.

2. Previous Work

The standard semantics for path programs without priorities results in sets of prefix-closed *vector-firing sequences* (VFS) [Shie79]. Equivalently, we can represent a VFS as *labeled partial order* (*lpo*), which yields the meaning of a process as a set of prefix-closed

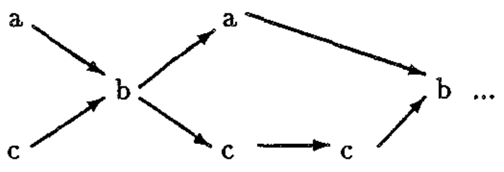


Figure 2: COSY behavior

lpo's. For priority path programs Janicki showed that partial orders, needed for expressing simultaneity as in the example in Figure 2, are inadequate. In the example given, the *simultaneous occurence* of a and c is one possible behavior, and so must be specified in the partial order shown. However, the priority constraint $b > c$ disallows the sequence ac. Yet the intended interpretation of the partial order implies ac is a possible observation, a violation of the priority constraint. It is worth noting that Shields made an initial attempt at extending the VFS to handle priority, and derived a class of priority path programs for which VFS is adequate [Shie81]. We shall not provide more details of the VFS semantics in this abstract: we shall use the *lpo* representation of VFS as a starting point for our semantics.

3. Prosset semantics for priority

3.1. The prosset model

In this section we review the *prosset* (preorder specification set) model of processes [Gaif87]. This model allows events to be partially related for both precedence and simultaneity. It is an extension of the *pomset* (partially ordered multiset) model developed by Pratt [Prat86]. The following definitions are from [Gaif87].

- **Definition.** A *multiset* is a structure $(\mathcal{V}, \Sigma, \mu)$ where $\mathcal{V}$ is the underlying set of events (vertices), Σ is the *alphabet* of *actions*, and $\mu : \mathcal{V} \rightarrow \Sigma$ labels the events with the actions. Each labeled event represents an occurence of an action. In the discussion below, it will be clear from context when we write u and mean a specific event with that label.

- **Definition.** A *behavior* is a structure $(\mathcal{V}, \Sigma, \Im, \mu)$ where $(\mathcal{V}, \Sigma, \mu)$ is a multiset and $\Im$ is a set of constraints on the events, interpreted conjunctively. For our purposes, we consider these constraints:

  $u < v$ (u precedes v)

$u \equiv v$ (u is simultaneous with v)

It is required that $<$ be irreflexive (ie., $(u < u) \not\in \Im$) and transitive. $\equiv$ must be an equivalence, and a congruence with respect to $<$ (ie., $u \equiv v < w \Rightarrow u < w$, $w < u \equiv v \Rightarrow w < v$). We can also define $u \leq v$ (u is not later than v) in terms of the other two relations: $u \leq v \Leftrightarrow u \equiv v$ or $u < v$. Dually, we could start with the primitives $\leq$ and $<$ and derive $\equiv$.

Prossets are behaviors with $<$ and either $\equiv$ or $\leq$. *Pomsets* are behaviors with only $<$. [1] A *linear prosset* is a prosset in which every element is related by either $\equiv$ or $<$: $<$ induces a total ordering on the equivalence classes of $\equiv$.
A *process* P is a class of behaviors closed under isomorphism with respect to a common alphabet. [2]

In what follows, we let P,Q,... denote sets of behaviors and p,q,... denote behaviors.

- **Definition.** Informally, a *prefix* is a subbehavior (ie., event subset that preserves the constraints) that includes all events preceeding an event in it, or simultaneous with an event in it. A $\preceq$*-prefix* is a prefix that may omit some simultaneous events.
Formally, p is a *subbehavior* of q, written $p \subset q$, if $\mathcal{V}_p \subseteq \mathcal{V}_q$, $\Sigma_p = \Sigma_q$, $\Im_p = \Im_q|_{\mathcal{V}_p}$ and $\mu_p = \mu_q|_{\mathcal{V}_p}$.
p is a *prefix* of q, written $p \sqsubseteq q$, if p is a subbehavior of q and for all $u, v \epsilon \mathcal{V}_p, \mathcal{V}_q$, $v \leq_q u \Rightarrow v \epsilon \mathcal{V}_p$.
Similarly, p is a $\preceq$*-prefix* of q, written $p \preceq q$, is defined using $v <_q p$.

- **Definition.** q is an *augment* of p, written $p \propto q$, if they have the same multisets and $\Im_p \subseteq \Im_q$, in other words if q adds more constraint arcs to p.
$\propto(P)$ denotes the set of augments of behaviors in P.

- **Definition.** A *linearization* of p is an augment of p which is linear (ie, total).
$\lambda(P)$ denotes the set of linearizations of behaviors in P.

Several points should be noted.

1. A linear behavior is equivalent to a sequence of sets of simultaneous events, as found in step semantics. We term a linear behavior an *observation*.

[1]Here, $<$ is a strict pre-order. It can be equivalently converted to a partial-order to correspond to pomsets as defined in [Prat86]. Note that $\leq$ is a preorder.

[2]We thus abstract away the underlying vertex set from the event structure. Taking the isomorphism class, those behaviors with bijections between event-sets preserving structure, was introduced by Gischer for pomsets [Gisc84]. There, a *pomset* is the isomorphism class of a labelled partial order.

2. Under the *intended interpretation* of unrelated events as implying all possible constraints, we would want each *behavior* to imply all possible *observations*. This is captured by requiring *augment closure*, where a process contains all the augments of its behaviors. [3]

3.2. Priority semantics by prosset refinement

Here we present a simple definition for the semantics of priority. For a given priority path program, we start out with the prefix-closed vector-firing sequences in partial-order form. This forms a set of prossets without simultaneity constraints, ie. pomsets. We then split out from the behaviors those observations which violate priority constraint but are implied from augments of the behavior. First, some definitions are in order:

- **Definition.** p is a *1-augment* of q, written $p = (q, u{\sim}v)$, when p is the same as q except for the one added constraint $u{\sim}v$ (ie, $\Im_p = \Im_q \cup (u{\sim}v)$).
  Here, $\sim$ is a meta-variable for some constraint, be it $\equiv$, $<$, or $\emptyset$ (no constraint).

- **Definition.** p is a *1-extension* of q, written $p = (q, u){\sim}v$, when p is the same as q except for the added event v and the relation $u{\sim}v$

- **Definition.** For meta-constraints $\sim$ and $\sim\prime$, $\sim$ *overlaps* $\sim\prime$ if $\sim\, = \,\sim\prime$ or $\sim\, = \emptyset$ or $\sim\prime = \emptyset$.

- **Definition.** Given a priority constraint $(y > x)$, we say
  p has a *choice conflict* at $(a{\sim}x)$ with q if

  $(p\prime, a){\sim}x$ is a prefix of p,

  $(q\prime, a){\sim}\prime y$ is a prefix of q,

  $q\prime \propto p\prime$ (ie, $p\prime$ is an augment of $q\prime$), and $\sim$ overlaps $\sim\prime$.

  The idea of choice conflict is that a prefix of p has a potentially enabled event in confict with another potentially enabled event with higher priority.

The definition of the priority semantics using a refinement relation is now straightforward. We want the behavior pomsets to be repeatedly split into several prossets until we remove all priority conflicts.

[$\Re$] Let P be a set of prossets.
We define the *split relation* $\Re \subseteq \propto (P)$ X $\propto (P)$ which removes one choice conflict as: $p \Re q$ iff p has a choice conflict with some $r \epsilon P$ at $a{\sim}x$, and q is a 1-augment of p at $a{\sim}x$ which resolves this conflict; q is $\emptyset$ if no resolution exists.

[3]This is similar to *pomset ideal* in [Gisc84].

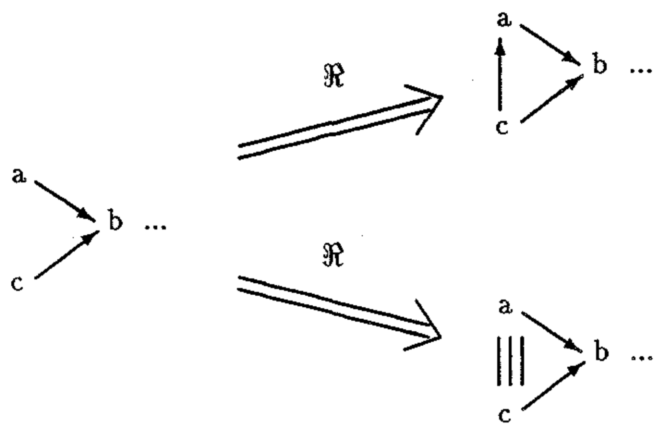


Figure 3: Behavior split into prossets

[$\mathcal{RS}$] We define the function which splits until all conflicts are resolved, as:
$\mathcal{RS}(P) = \{p : \exists q \epsilon P \text{ s.t. } q\Re^* p, \text{ and } \nexists r : p\Re^+ r\}$.

For infinite splits, we can use *limits* to define $\mathcal{RS}$:

- A $\propto$-*chain* of behaviors is a set of behaviors totally ordered by $\propto$.
  If $(p_i)_{i\epsilon I}$ is a $\propto$-chain, $\propto$-*lim* p_i is the $\propto$-maximal behavior p such that $p_i \propto p$ for all $i\epsilon I$. [4]

Note that an $\Re$-chain is also an $\propto$-chain. Then the split function becomes:

[$\mathcal{RS}$] $\mathcal{RS}(P) = \{q : q = \propto\text{-lim } p_i$ for some $\Re$-chain p_i in $P\}$.

The meaning of a priority path program *Prog*, given its set of vector-firing sequences $VFS(Prog)$ in pomset form, is then:
$\mathcal{RS}(VFS(Prog))$.

Figure 3 shows one split in the behavior of figure 2 to remove the *ac* observation.

3.3. Priority semantics by maximal ideal partitions

In this section we briefly give a variant of the previous semantics. Again we require a few more definitions.

[4]In a dual fashion, we may define $\subset$-lim as the $\subset$-minimal behavior for a $\subset$-chain.

- An *augment-consistent* set of prossets P is one where all behaviors are augments of a common behavior. [5]

- Given an (augment) consistent set of prossets $(p_i)_{i \in I}$, define $\cap(p_i) = (\cup\mathcal{V}(p_i),\ \Sigma,\ \cap\Im(p_i),\ \cup\mu(p_i))$. This is the maximal behavior contained in all the prossets: it represents a maximal order which is satisfied by every observation. [6] Note that $\propto\text{-}lim\ p_i = \cap(p_i)$.

- Define P is *ideal* iff P is augment-consistent and $\lambda(\cap(P)) = \lambda(P)$. This means that a history may be equivalently represented as one partial order, in the sense that it provides the same observations. This roughly corresponds to Janicki's definition of ideal histories.

- A partition of a set of prossets is an *ideal partition* iff each subset in the partition is ideal.

- A partition is a *coarsening* of another if its subsets are unions of the others'.

- A *maximal ideal partition* of P is a maximal element over the coarsening ordering on ideal partitions of P. Let $\mathcal{I}(P) = \{$ maximal ideal partitions of $P\}$.

Then the maximal ideal semantics for priority is given by reducing to linear behaviors, eliminating those conflicting priority constraints, and coming back up to the maximal prosset level. Formally, we have

$[\mathcal{MS}]\ \mathcal{MS}(P) = \{\cap(Q) : Q \epsilon \mathcal{I}(\mathcal{RS}(\lambda(VFS(Prog))))\}$

Note that the splitting relation applied to linear behaviors will merely delete it if priority conflict exists.

3.4. Relation to sequence of multiple semantics

Janicki's multiple firing sequence semantics is equivalent to the linearization of either of our theories. This is expressed in the commutative diagram in Figure 4.
In Janicki's semantics, the meaning of a process is a set of "sequence of maximal antichains", a sequence of simultaneous actions, that is equivalent to a linear prosset.

[5] A *consistent* set of prossets P is one whose events are described by the same vertex set, ie., for all p, q in P, $\mu_p|_{\cap(\mathcal{V}_p,\mathcal{V}_q)} = \mu_q|_{\cap(\mathcal{V}_p,\mathcal{V}_q)}$.

[6] Dually, we may define $\cup(P)$ as the minimal behavior.

$$\begin{tikzcd}
\text{COSY} \arrow[r] \arrow[rrrdd, "\text{Janicki's sequences of multiples}"'] & \text{VFS} \cong \text{Pomset} \arrow[r, "\mathcal{RS}"] \arrow[rd, "\mathcal{MS}"'] & \text{Prossets} \arrow[r, "\lambda"] & \text{Linear Behaviors} \arrow[dd, equal] \\
 & & \text{Prossets} \arrow[ru, "\lambda"'] & \\
 & & & \text{Multiple Firing Sequences}
\end{tikzcd}$$

Figure 4: Equivalence of semantics

4. Taxonomy

We briefly consider a simple taxonomy relating the various semantics to other event-history models. Within the realm of the semantics dealt with in this paper, we have the diagram shown in Figure 5.

In the larger realm, following the examples of [Rutt89, Reis89, Shie89], we can categorize event-history models according to two dimensions. The first dimension is that of **Behavior Structures**, which may be

- *branching-time* (multi-tree),
- *linear-time* (full-path), or
- *hybrid* (stubbed partial-path).

The second dimension is that of **Constraints among Events** [Shie89], which include

- *precedence* (causality),
- *concurrency* (unconstrained over causality),
- *simultaneity*, and
- *conflict* (or enabling).

Models may have several combinations of constraints. We use the terms *partial-order precedence* for precedence and concurrency, and *linear-order precedence* (interleaving) for systems

$$\begin{tikzcd}
 & \text{Step semantics} \cong \text{Linear prossets} \arrow[dr, ">"] & \\
\text{Prosset} \arrow[ur, ">"] \arrow[dr, ">"'] & & \text{Interleaving} \\
 & \text{Pomsets} \arrow[ur, ">"'] &
\end{tikzcd}$$

Figure 5: Relation of these semantics

having precedence and no concurrency. For classification purposes, we will use a slightly altered second dimension, replacing "precedence" and "concurrency" by the just mentioned two categories. *True concurrency* includes both partial-order precedence and simultaneity categories.

As an example, prossets fall into linear-time simultaneity with partial-order precedence, while step semantics fall into linear-time simultaneity with linear-order precedence.

For a deeper analysis of categorizing behavioral presentations by types of event relations, refer to [Shie89]. For an alternative formulation differentiating precedence and causality, as well as abstracting away the underlying time-model (point, interval, relative, etc.) from the description of concurrency and simultaneity, see [Gaif89].

5. Summary and Future Work

We presented a semantics for priority using behaviors with partial precedence and simultaneity relations. It is similar in concept to previous denotations for priority, but is recast into a more elegant theory.

In the immediate future, the issues of fairness and infinite streams need to be addressed for the priority semantics. Applications and comparisons must be made with the fairness results of Haim Gaifman for prossets and those of Janicki for multiples semantics. A question to consider is whether the prossets semantics is equivalent in power to sequences of simultaneous actions, or does it provide some distinguishing power for a property of interest? It seems possibly not, since the latter still reduces concurrency to interleaving.

Acknowledgements

This research was supported in part by the National Science Foundation under grant number DCR-8406850.

References

[Best86] Best, E., "COSY: Its Relation to Nets and CSP", *LNCS 255*, Springer-Verlag (1987) pp. 416-440.

[Camp74] Campbell, R.H and Haberman, A.N., "The specification of process synchronisation by paths expressions", *LNCS 16*, Springer-Verlag (1974) pp.89-102.

[Camp76] Campbell, R.H., *Path Expressions: a technique for process synchronization*, Ph.D. Thesis, University of Newcastle upon Tyne (1976).

[Gisc84] Gischer, J., *Partial Orders and the Axiomatic Theory of Shuffle*, Ph.D. thesis, Computer Science Dept., Stanford University (December, 1984).

[Gaif87] Gaifman, H. and Pratt, V.R., "Partial Order Models of Concurrency and the Computation of Functions", *Proceedings of the Second Symposium on Logic in Computer Science* (Ithica, N.Y., June, 1987), IEEE, New York, 1987, pp. 72-85.

[Gaif89] Gaifman, Haim, "Modeling concurrency by partial orders and nonlinear transition systems", *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (LNCS 354)*, J.W.de Bakker, W.-P.de Roever and G.Rozenberg (eds), Springer-Verlag (1989) pp.467-488.

[Inmos87] INMOS Ltd., *The occam programming manual*, Prentice Hall (1987).

[Jani87] Janicki, Ryszard, "A Formal Semantics for Concurrent Systems with a Priority Relation", *Acta Informatica 24*, (1987) pp.33-55.

[Jani88] Janicki, Ryszard and Lauer, Peter E., "On the semantics of priority systems", *Proceedings of the International Conference on Parallel Processing*, 1988, pp. 150-156.

[Lamp84] Lamport, Leslie, "What it means for a concurrent program to satisfy a specification: Why no one has specified priority", *3rd ACM Symposium on Principles of Distributed Computing*, (Vancouver, 1984) pp. 78-83 (February, 1986) pp. 33-71.

[Laue75] Lauer, P.E. and Campbell, R.H., "Formal Semantics for a Class of High-level Primitives for Coordinating Concurrent Processes", *Acta Informatica 5*, (1975) pp. 297-332.

[Laue79] Lauer, P.E., Torrigiani, P.R. and Shields, M.W., "COSY: A System Specification Language based on Paths and Processes", *Acta Informatica 12*, (1979) pp. 109-158.

[Meye89] Myer, J.-J.C. and de Vink, E.P., "Step semantics for "true" concurrency with recursion", *Distributed Computing*, Vol.3, No.3 (1989) pp. 130-145.

[Prat86] Pratt, V.R., "Modelling Concurrency with Partial Orders", *International Journal of Parallel Programming*, Vol.15, No.1 (February, 1986) pp. 33-71.

[Quir85] Quirk, W.J. (ed), *Verification and Validation of Real-time Software*, Springer-Verlag (1985).

[Reis85a] Reisig, W., "On the Semantics of Petri Nets", in: *Formal Models in Programming*, E.J.Neuhold and G.Chroust (eds.), IFIP (1985) pp. 347-372

[Reis85b] Reisig, W., *Petri nets*, Springer-Verlag (1985).

[Reis89] Reisig, W., "Towards a Temporal Logic for Causality and Choice in Distributed Systems", *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (LNCS 354)*, J.W.de Bakker, W.-P.de Roever and G.Rozenberg (eds), Springer-Verlag (1989) pp.603-627.

[Roze83] Rozenberg, G. and Verraedt, R., "Subset Languages of Petri Nets, Parts I and II", *TCS 26*, (1986) pp.301-326, and *TCS 27*, (1986) pp.85-108.

[Rutt89] Rutten, J.J.M.M, "Correctness and full abstraction of metric semantics for concurrency", *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (LNCS 354)*, J.W.de Bakker, W.-P.de Roever and G.Rozenberg (eds), Springer-Verlag (1989) pp.628-659.

[Shie79] Shields, M.W., "Adequate Path Expressions", *LNCS 70*, Springer-Verlag (1979) pp.249-265.

[Shie81] Shields, M.W., "On the Non-sequential Behavior of Systems Possessing a General Free-choice Property", *Report CRS-92-81*, Dept. of Computer Science, University of Edinburgh (1981).

[Shie89] Shields, M.W., "Behavioral Presentations", *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency (LNCS 354)*, J.W.de Bakker, W.-P.de Roever and G.Rozenberg (eds), Springer-Verlag (1989) pp.673-689.

[Taub87] Taubner, Dirk and Vogler, Walter, "The Step Failure Semantics", in: *Proc. STACS 87, LNCS 247*, F.J.Brandenburg,and N.G.Vidal (eds), Springer-Verlag (1987) pp. 348-359.